A reflected feature space for CART

D. C. Wickramarachchi[1], B. L. Robertson[2*], M. Reale[2], C. J. Price[2] and J. A. Brown[2]

University of Sri Jayewardenepura and the University of Canterbury

Summary

This paper presents an algorithm for learning oblique decision trees, called HHCART(G). Our decision tree combines learning concepts from two classification trees, HHCART and Geometric Decision Tree (GDT). HHCART(G) is a simplified HHCART algorithm that uses linear structure in the training examples, captured by a modified GDT angle bisector, to define splitting directions. At each node, we reflect the training examples with respect to the modified angle bisector to align this linear structure with the coordinate axes. Searching axis parallel splits in this reflected feature space provides an efficient and effective way of finding oblique splits in the original feature space. Our method is much simpler than HHCART because it only considers one reflected feature space for node splitting. HHCART considers multiple reflected feature spaces for nodes splitting making it more computationally intensive to build. Experimental results show that HHCART(G) is an effective classifier, producing compact trees with similar or better results than several other decision trees, including GDT and HHCART trees.

Key words: decision trees; geometric decision tree; multi-class classification; oblique decision tree.

1. Introduction

Decision tree classifiers are conceptually simple, making them a popular statistical learning method. To build a decision tree classifier, the feature space is recursively divided into disjoint regions until each region is predominately of one class. This partitioning is displayed in a tree structure called a decision tree, with the root node at the top of the tree. Each internal node (including the root node) of the tree has an associated splitting rule, which is a function of the features present at that node. Leaf nodes have no descent nodes and have

*Author to whom correspondence should be addressed.
[1] Department of Statistics, University of Sri Jayewardenepura, Gangodawila, Nugegoda, Sri Lanka
[2] School of Mathematics and Statistics, University of Canterbury, Private Bag 4800, Christchurch, New Zealand
Email: blair.robertson@canterbury.ac.nz

Acknowledgment.

an associated class label. To classify a feature vector, the splitting rules are used to determine a unique path through the tree from the root node to a leaf node. The feature vector is then assigned the class label of the majority class at that leaf node.

In this paper, we consider binary trees with linear splitting rules for continuous feature vectors. In binary trees, each nonleaf node is split into two descendant nodes. They are popular because non-binary trees tend to fragment the training examples quickly, leaving too few training examples at nodes further down the tree (Hastie, Tibshirani & Friedman 2013, Section 9.2). If each splitting rule is a function of a single feature, the resulting tree is called axis parallel (Breiman et al. 1984). Otherwise, the tree is called oblique and each splitting rule is a linear combination of the available features (Breiman et al. 1984). A popular method for building axis parallel trees is called Classification and Regression Trees (CART) (Breiman et al. 1984). CART trees are simple and easy to interpret, but oblique trees tend to have fewer leaf nodes and have better accuracy (Li, Dong & Kothari 2005). However, oblique trees are less interpretable and finding an optimal split can be computationally demanding (Wickramarachchi et al. 2016; Hastie, Tibshirani & Friedman 2013, Section 9.2).

Many oblique decision trees have been presented in the statistical learning literature. One of the first was Classification and Regression Trees Linear Combination (CART-LC) (Breiman et al. 1984). This algorithm uses a deterministic hill-climbing method followed by a backward elimination process to search for the best split with respect to an impurity function. Simulated Annealing Decision Tree (SADT) also searches for the best split, but it uses the simulated annealing optimisation algorithm (Heath, Kasif & Salzberg 1993). However, a large number of candidate splits may need to be tested before the best split is found (Murthy, Kasif & Salzberg 1994). The concepts of CART-LC and SADT were combined to create a system of decision tree induction algorithms called OC1 (Murthy, Kasif & Salzberg 1994). The OC1 system has also been extended to include genetic optimisation algorithms (Cantú-Paz & Kamath 2003) and differential evolution-based optimisation algorithms (Rivera-Lopez et al. 2017) to improve the search. Rather than using optimisation algorithms to search for the best split, splits can be based on statistical techniques (Gama & Brazdil 1999; Li et al. 2003; Kolakowska & Malina 2005; Truong 2009; López-Chau et al. 2013; Sheikholharam Mashhadi 2018) or on structural heuristics of the decision boundary (Amasyah & Ersoy 2008; Manwani & Sastry 2012). Instead of recursively partitioning the feature space using locally optimal splits, Bertsimas & Dunn (2017) build the entire tree at once using techniques from mixed-integer optimisation. Their approach is motivated by the fact that recursive partitioning methods use splits that are not guided by the possible influence of future splits. They show that their approach is tractable on a number datasets and that it performs better than CART.

The remainder of this paper is organised as follows. The following section describes two decision trees, Geometric Decision Tree (GDT; Manwani & Sastry 2012) and HHCART (Wickramarachchi et al. 2016), that are pertinent to this paper. Section 3 introduces a modified GDT angle bisector that is used to extend HHCART to produce a new classification tree, called HHCART(G). Section 4 describes HHCART(G) and experimental results are presented in Section 5. Concluding remarks are given in Section 6.

2. Related Decision Trees

Geometric Decision Tree (GDT; Manwani & Sastry 2012) is an oblique decision tree that attempts to capture linear tendencies in the training examples, rather than searching for the best split with respect to an impurity function. Consider splitting a node t for a two-class classification problem. First, two clustering hyperplanes are formed (one for each class), where each hyperplane is in some sense closest to all points of one class and is farthest from all points of the other class (Manwani & Sastry 2012). The angle bisectors between the clustering hyperplanes are then formed and the bisector with the lowest Gini index is chosen for the split. For multi-class classification problems, the training examples at node t are divided in two subsets. The first subset is the majority class at node t and the second subset is the remaining training examples at node t. The split is then found using these two subsets as described above. The method is recursively run on all descendent nodes until the impurity at each leaf node is sufficiently small.

HHCART (Wickramarachchi et al. 2016) is another oblique decision tree. Rather than searching for oblique splits directly, HHCART finds the best axis parallel split in a set of reflected feature spaces. This split will be oblique in the original feature space. Consider splitting a node t containing p quantitative features and C classes. Each reflected feature space at node t is defined using a Householder matrix (Robertson, Price & Reale 2013):

$$\boldsymbol{H} = \boldsymbol{I} - 2\frac{(\boldsymbol{e} - \boldsymbol{d}_{ik})(\boldsymbol{e} - \boldsymbol{d}_{ik})^\top}{\|\boldsymbol{e} - \boldsymbol{d}_{ik}\|^2}, \tag{1}$$

where $\boldsymbol{e}$ is the first column of the p-dimensional identity matrix $\boldsymbol{I}$ and $\boldsymbol{d}_{ik}$ is the ith unit scaled eigenvector of the estimated covariance matrix of class k examples at node t. Each feature vector $\boldsymbol{x}$ (column vector) at node t is reflected using $\boldsymbol{Hx}$ and the best axis parallel split in the reflected training examples is found. Reflecting the feature vectors in this way makes $\boldsymbol{d}_{ik}$ parallel to $\boldsymbol{e}$ and provides a simple and effective way to find oblique splits (Robertson, Price & Reale 2013; Wickramarachchi et al. 2016). The authors propose two HHCART methods. HHCART(A) considers all Cp reflected feature spaces defined using (1) and chooses the axis parallel split that maximises the reduction in impurity. HHCART(D) only considers

C reflected spaces at each node, defined using the dominant eigenvector of the estimated covariance matrix for each class. The method is recursively run on all descendent nodes until no further splitting is possible. The authors recommend growing a large HHCART tree, which is then pruned using cost-complexity pruning (Breiman et al. 1984). HHCART(A) has slightly better accuracy than HHCART(D), but HHCART(D) is simpler and is more computationally efficient to build (Wickramarachchi et al. 2016).

In this paper, we present an oblique classification tree called HHCART(G). Our tree uses a modified GDT angle bisector to define an alternative reflected feature space for the HHCART algorithm. This reflected feature space aligns linear structure in the training examples captured by the angle bisector with the coordinate axes. Searching axis parallel splits in this reflected feature space provides an efficient way of finding effective oblique splits in the original space. HHCART(G) is much simpler than the other HHCART trees because it only considers one reflected feature space for node splitting. Because we only search one feature space, the computational effort required to build an HHCART(G) tree is substantially less than the other HHCART trees. To split each node in HHCART(D) (the more computationally efficient HHCART algorithm) we solve C generalised eigenvalue problems, reflect the training examples C times and search Cp splitting dimensions for the best split (Wickramarachchi et al. 2016). To split a node using HHCART(G), we solve one generalised eigenvalue problem, reflect the training examples once and search p splitting dimensions for the best split.

3. The modified angle bisector

In this section, we define the modified angle bisector that HHCART(G) uses to define its reflected feature space. This vector was introduced in Manwani & Sastry (2012) and its derivation is summarised here for convenience. The interested reader is referred to Manwani & Sastry (2012) for full details. Our modified angle bisector handles specific rank deficient matrices resulting from small sample sizes that were not covered in Manwani & Sastry (2012).

Consider a two-class classification problem with training examples

$$D = \{(\boldsymbol{x}_i, y_i) : \boldsymbol{x}_i \in \mathbb{R}^p, y_i \in \{-1, 1\} \text{ and } i = 1, 2, \ldots, n\}.$$

Let $\boldsymbol{A} \in \mathbb{R}^{n_A \times p}$ be the matrix containing points with $y_i = 1$ at node t. Similarly, let $\boldsymbol{B} \in \mathbb{R}^{n_B \times p}$ be the matrix containing points with $y_i = -1$ at node t. The angle bisector

is defined using two clustering hyperplanes (one for each class), each of the form

$$\boldsymbol{w}^\top \boldsymbol{x} + b = 0.$$

For convenience, we write $\widetilde{\boldsymbol{w}} = (\boldsymbol{w}^\top, b)^\top$ and $\widetilde{\boldsymbol{x}} = (\boldsymbol{x}^\top, 1)^\top$ so that $\boldsymbol{w}^\top \boldsymbol{x} + b = \widetilde{\boldsymbol{w}}^\top \widetilde{\boldsymbol{x}}$. The clustering hyperplanes are chosen so that, in some sense, each hyperplane is closest to all points of one class and is farthest from all points of the other class. Specifically, they are solutions to the following optimisation problems

$$\widetilde{\boldsymbol{w}}_1 = \text{argmax}_{\widetilde{\boldsymbol{w}} \neq 0} \frac{\widetilde{\boldsymbol{w}}^\top \boldsymbol{M} \widetilde{\boldsymbol{w}}}{\widetilde{\boldsymbol{w}}^\top \boldsymbol{G} \widetilde{\boldsymbol{w}}} \quad \text{and} \quad \widetilde{\boldsymbol{w}}_2 = \text{argmin}_{\widetilde{\boldsymbol{w}} \neq 0} \frac{\widetilde{\boldsymbol{w}}^\top \boldsymbol{M} \widetilde{\boldsymbol{w}}}{\widetilde{\boldsymbol{w}}^\top \boldsymbol{G} \widetilde{\boldsymbol{w}}}, \tag{2}$$

where

$$\boldsymbol{M} = \frac{1}{n_B} (\boldsymbol{B}, \mathbf{1})^\top (\boldsymbol{B}, \mathbf{1}) \quad \text{and} \quad \boldsymbol{G} = \frac{1}{n_A} (\boldsymbol{A}, \mathbf{1})^\top (\boldsymbol{A}, \mathbf{1}),$$

and $\mathbf{1}$ is a column vector of ones. If matrix $\boldsymbol{G}$ has full column rank, the solutions to these optimisation problems can be obtained using a LU-decomposition method (Golub & van Loan 1996) and they are the eigenvectors corresponding to the maximum and minimum eigenvalues from the following generalised eigenvalue problem (Manwani & Sastry 2012)

$$\boldsymbol{M} \widetilde{\boldsymbol{w}} = \lambda \boldsymbol{G} \widetilde{\boldsymbol{w}}. \tag{3}$$

Let $\widetilde{\boldsymbol{w}}_1 = (\boldsymbol{w}_1^\top, b_1)^\top$ and $\widetilde{\boldsymbol{w}}_2 = (\boldsymbol{w}_2^\top, b_2)^\top$ be the eigenvectors corresponding to the maximum and minimum eigenvalues from (3), respectively, scaled so that $\|\boldsymbol{w}_1\| = \|\boldsymbol{w}_2\| = 1$. If $\boldsymbol{w}_1 = \boldsymbol{w}_2$, the angle bisector is

$$\widetilde{\boldsymbol{w}}_3 = (\boldsymbol{w}_1^\top, (b_1 + b_2)/2)^\top. \tag{4}$$

Otherwise, $\boldsymbol{w}_1 \neq \boldsymbol{w}_2$ and the angle bisectors are

$$\widetilde{\boldsymbol{w}}_3 = \widetilde{\boldsymbol{w}}_1 + \widetilde{\boldsymbol{w}}_2 \quad \text{and} \quad \widetilde{\boldsymbol{w}}_4 = \widetilde{\boldsymbol{w}}_1 - \widetilde{\boldsymbol{w}}_2. \tag{5}$$

Both of these hyperplanes are evaluated at node t using the hyperplane Gini index

$$\text{Gini}(\widetilde{\boldsymbol{w}}) = 2L(1 - L_A)L_A + 2(1 - L)(1 - R_A)R_A, \tag{6}$$

where L is the fraction of points at the left descendent node using split $\widetilde{\mathbf{w}}$ and L_A (R_A) is the fraction of points at the left (right) node from matrix $\boldsymbol{A}$. The angle bisector with the lower Gini index is chosen to split the node (Manwani & Sastry 2012).

3.1. Small sample sizes

As decision trees grow, fewer and fewer training examples are available for splitting internal nodes. This can cause matrix $\boldsymbol{G}$ to become rank deficient (matrix $\boldsymbol{A}$ does not have full column rank) and hence, the LU-decomposition based method described above is not applicable. In this case, Manwani & Sastry (2012) define the angle bisector as the eigenvector corresponding to the largest eigenvalue of

$$\widetilde{\boldsymbol{M}} = \boldsymbol{Q}\boldsymbol{Q}^\top \boldsymbol{M} \boldsymbol{Q}\boldsymbol{Q}^\top,$$

where $\boldsymbol{Q}$ is a matrix whose columns are an orthonormal basis for the null($\boldsymbol{G}$). However, if null($\boldsymbol{G}$) $\subseteq$ null($\boldsymbol{M}$), $\widetilde{\boldsymbol{M}}$ will be the zero matrix and the method fails. Wickramarachchi (2015) provides a remedy for this special case.

Define $\widetilde{\boldsymbol{w}} = \boldsymbol{u} + \boldsymbol{v}$, where $\boldsymbol{u} \in$ range($\boldsymbol{G}$) and $\boldsymbol{v} \in$ null($\boldsymbol{G}$). Then the left optimisation in (2) becomes (Wickramarachchi 2015)

$$\widetilde{\boldsymbol{w}}_1 = \text{argmax}_{\widetilde{\boldsymbol{w}} \neq 0} \frac{(\boldsymbol{u}+\boldsymbol{v})^\top \boldsymbol{M} (\boldsymbol{u}+\boldsymbol{v})}{(\boldsymbol{u}+\boldsymbol{v})^\top \boldsymbol{G} (\boldsymbol{u}+\boldsymbol{v})} = \text{argmax}_{\boldsymbol{u} \neq 0} \frac{\boldsymbol{u}^\top \boldsymbol{M} \boldsymbol{u}}{\boldsymbol{u}^\top \boldsymbol{G} \boldsymbol{u}},$$

because $\boldsymbol{v} \in$ null($\boldsymbol{G}$) with null($\boldsymbol{G}$) $\subseteq$ null($\boldsymbol{M}$) and values of $\boldsymbol{u}$ giving $\boldsymbol{u}^\top \boldsymbol{G} \boldsymbol{u} = 0$ are avoided. The right optimisation in (2) is solved in a similar way. Define $\tilde{\boldsymbol{w}} = \boldsymbol{u} + \boldsymbol{v}$, where $\boldsymbol{u} \in$ range($\boldsymbol{M}$) and $\boldsymbol{v} \in$ null($\boldsymbol{M}$). Then, the right optimisation in (2) becomes (Wickramarachchi 2015)

$$\widetilde{\boldsymbol{w}}_2 = \text{argmax}_{\widetilde{\boldsymbol{w}} \neq 0} \frac{(\boldsymbol{u}+\boldsymbol{v})^\top \boldsymbol{G} (\boldsymbol{u}+\boldsymbol{v})}{(\boldsymbol{u}+\boldsymbol{v})^\top \boldsymbol{M} (\boldsymbol{u}+\boldsymbol{v})} = \text{argmax}_{\boldsymbol{u} \neq 0} \frac{\boldsymbol{u}^\top \boldsymbol{G} \boldsymbol{u}}{\boldsymbol{u}^\top \boldsymbol{M} \boldsymbol{u}},$$

because $\boldsymbol{v} \in$ null($\boldsymbol{M}$) with null($\boldsymbol{M}$) $\supseteq$ null($\boldsymbol{G}$) and values of $\boldsymbol{u}$ giving $\boldsymbol{u}^\top \boldsymbol{M} \boldsymbol{u} = 0$ are avoided. The angle bisectors of $\widetilde{\boldsymbol{w}}_1$ and $\widetilde{\boldsymbol{w}}_2$ are calculated using (5) and the hyperplane with the lower hyperplane Gini index (6) is chosen to split the node. If $\widetilde{\boldsymbol{w}}_1$ and $\widetilde{\boldsymbol{w}}_2$ are parallel, the angle bisector is (4).

4. HHCART(G) Algorithm

The HHCART(G) algorithm for splitting a single node is given in Algorithm 1 and is illustrated in the following subsection. The algorithm has two user-defined parameters, which are $n_{\min}$, the minimum number of training examples and ϵ, the minimum node impurity. We recommend using 10-fold cross-validation to choose ϵ as suggested in Manwani & Sastry (2012). If there are fewer than $n_{\min}$ training examples or if the node's Gini index is less than

Algorithm 1: The HHCART(G) algorithm at a single node

Data: Training examples at node t, $D^t = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$.
Initialise: Choose $n_{\min}$ and ϵ.
Divide the points from D^t into two matrices:
Matrix $\boldsymbol{A} \in \mathbb{R}^{n_A \times p}$ contains points from the majority class; and
Matrix $\boldsymbol{B} \in \mathbb{R}^{n_B \times p}$ contains points from the remaining classes.
if $n > n_{\min}$ *and Gini*$(t) > \epsilon$ **then**
Compute the angle bisector $\boldsymbol{w}$ (unit scaled) using $\boldsymbol{A}$ and $\boldsymbol{B}$ (see Section 3).
Construct the Householder matrix

$$\boldsymbol{H} = \boldsymbol{I} - 2\frac{(\boldsymbol{e}-\boldsymbol{w})(\boldsymbol{e}-\boldsymbol{w})^\top}{\|\boldsymbol{e}-\boldsymbol{w}\|^2}, \tag{7}$$

where $\boldsymbol{e}$ is the first column of $\boldsymbol{I}$, the identity matrix.
Reflect the training examples, $\widehat{D}^t = \{\boldsymbol{H}\boldsymbol{x}_i, y_i\}_{i=1}^n$.
Find the best axis parallel split in $\widehat{D}^t$, $z_j \leq s^t$, where z_j is the jth coordinate direction in the reflected feature space.
Return the jth column of $\boldsymbol{H}$ and s^t.
else
Node t is a leaf node.
end

ϵ, node t is a leaf node whose class label is the node's majority class. Otherwise, the impure node is split.

To split node t, the angle bisector from Section 3 is used to define a reflected feature space. The training examples at node t are reflected using (7) so that the normal vector of the angle bisector $\boldsymbol{w}$ is parallel to $\boldsymbol{e}$, the first coordinate axis. Using the reflected training examples $\widehat{D}^t$, we find the axis parallel split s^t that minimises the split Gini index

$$\text{Gini}(s) = L\sum_{k=1}^{C} p_{lk}(1-p_{lk}) + (1-L)\sum_{k=1}^{C} p_{rk}(1-p_{rk}),$$

where L is the fraction of points that go to the left descendent node after the split and p_{lk} (p_{rk}) is the fraction of points at the left (right) node from class k. The best split is found by exhaustively searching over all p dimensions, which can be done efficiently because each dimension can be treated separately and the search is embarrassingly parallel. Let $z_j \leq s^t$ (the jth coordinate direction in the reflected feature space) be the best axis parallel split. In the original feature space, this split is oblique and given by $\boldsymbol{h}_j^\top \boldsymbol{x} \leq s^t$, where $\boldsymbol{h}_j$ is the jth column of $\boldsymbol{H}$. The method is recursively run on all descendent nodes until no further splitting is possible.

Table 1. Datasets with quantitative features, downloaded from UCI repository.

Dataset	Features	Classes	Examples
Balance Scale (BS)	4	3	625
Boston Housing (BH)	13	2	506
Breast Cancer (BC)	9	2	638
BUPA	6	2	345
Glass (GLS)	9	7	214
Pima Indian (PIND)	8	2	768
Wine (WINE)	13	3	178
Survival (SUR)	3	2	306
Heart (HRT)	13	2	270
Letter (LET)	16	26	20,000

4.1. Illustrative Example

In this subsection, we illustrate the HHCART(G) algorithm for a simple three-class classification problem. The classes have bivariate Gaussian distributions, with different means and different variance covariance matrices. Training samples from each class are shown in Figure 1(a).

Initially, classes 1 and 2 are combined to define a binary classification problem. The clustering hyperplanes (dashed) and the angle bisector $\boldsymbol{w}$ for this binary problem are shown in Figure 1(b). The training observations are then reflected using (7) so that the normal vector of $\mathbf{w}$ is set parallel to $\boldsymbol{e}_1$. The best axis parallel split for the reflected training examples, $\boldsymbol{z}_1 \leq -1.88$, is shown in Figure 1(c). In the original feature space, this split is oblique and given by $0.76x_1 + 0.65x_2 \geq 1.88$ (see Figure 1(f)). The method then repeats on the impure right daughter node, by forming the reflected feature space and finding the best axis parallel split (Figure 1(d,e)). The final tree has two splits and is shown in Figure 1(f).

5. Experimental Results and Discussion

In this section, we compare HHCART(G) with Geometric Decision Tree (GDT) and the results from Wickramarachchi et al. (2016), which includes results for the HHCART algorithms and the OC1 methods (Murthy, Kasif & Salzberg 1993). OC1-AP and OC1-LC are implementations of CART and CART-LC (Breiman et al. 1984) in the OC1 system, but OC1-LC does not include CART-LC's backward elimination procedure (Wickramarachchi et al. 2016). Experiments were performed using datasets from Bache & Lichman (2013) (see Table 1) so we could make comparisons with existing results. Our experimental setup was also the same as in Wickramarachchi et al. (2016). We used ten 5-fold cross-validations to

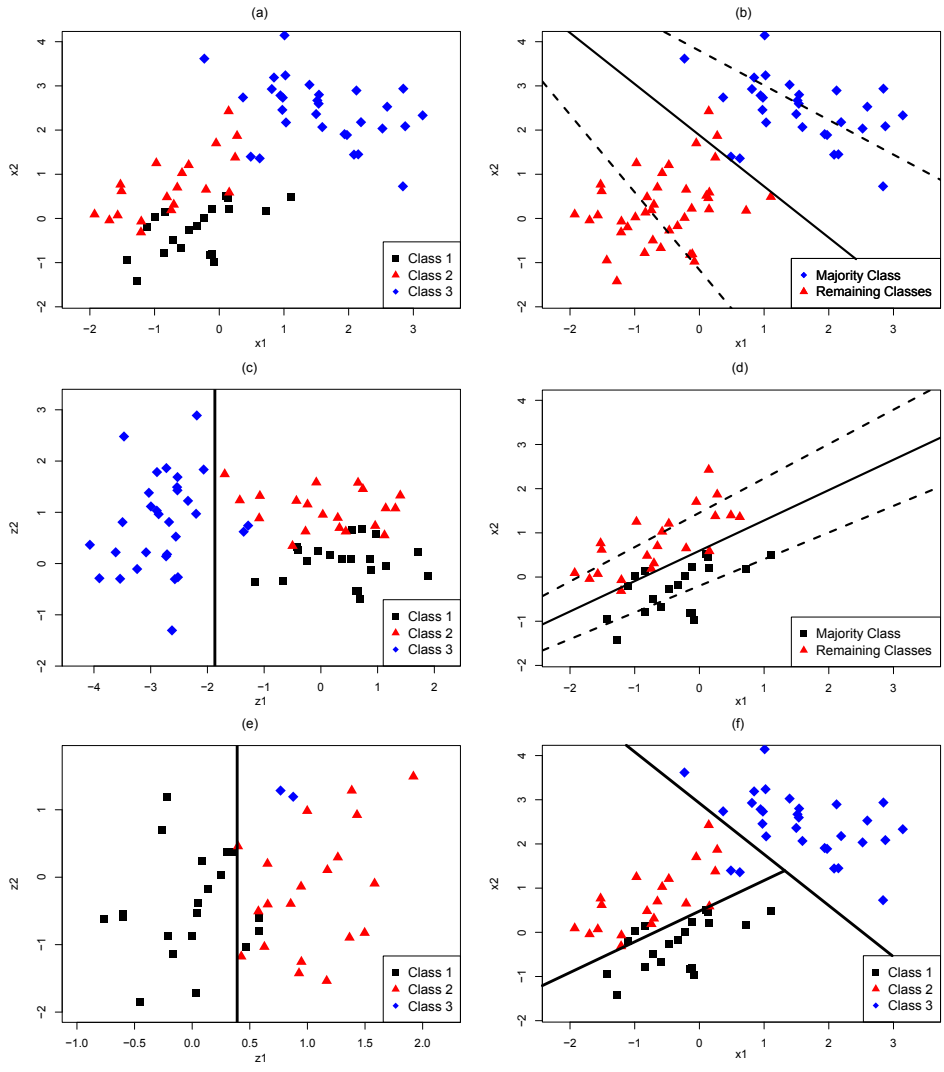

Figure 1. (a) Training Data. (b) Clustering hyperplanes (dashed) and the angle bisector. (c) Reflected training examples and the best axis parallel split. (d) Clustering hyperplanes (dashed) and the angle bisector for the right daughter node. (e) Reflected training examples and the best axis parallel split. (f) Fully grown HHCART(G) tree.

estimate the accuracy and the size (number of leaf nodes) of each tree. For HHCART(G), $n_{\min} = 2$ and ϵ was chosen using 10-fold cross-validation as in Manwani & Sastry (2012). The GDT algorithm that we used was slightly different than the original algorithm because it

included our remedy for rank deficient matrices (see Subsection 3.1). GDT only has one user-defined parameter (ϵ, minimum node impurity), which was estimated using 10-fold cross-validation (Manwani & Sastry 2012).

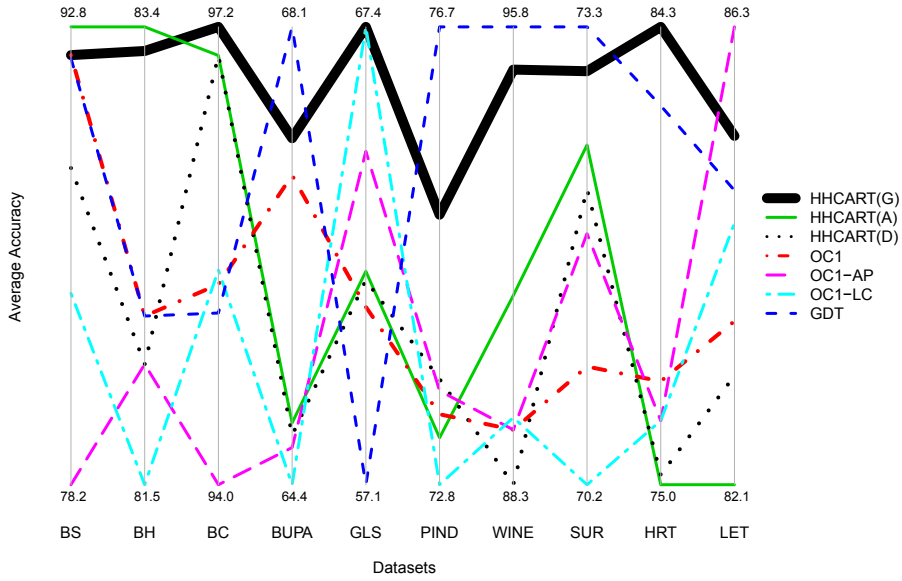

Figure 2. A parallel coordinates plot of the average accuracy from ten 5-fold cross-validations for seven decision tree methods. The maximum and minimum average accuracies for each dataset are also reported. The HHCART(G) line is in bold.

The results from our experiment are illustrated in Figure 2 and reported in Table 2 along with the respective standard deviations (computed over ten runs). Following Manwani & Sastry (2012), we say that an algorithm is significantly better than another if its reported average is at least one standard deviation better than the other.

The average accuracy of HHCART(G) was better than (or equal to) all of the other decision trees on test problems BC, GLS and HRT. For the remaining seven problems, HHCART(G) was the second best decision tree. The only algorithm that gave an average accuracy that was significantly better than HHCART(G) was OC1-AP on the LET dataset.

HHCART(G) was significantly better than GDT on BC, GLS, HRT and LET and GDT was significantly better on PIND. For the remaining four problems, there was no significant difference between HHCART(G) and GDT. The average tree size of HHCART(G) was significantly smaller than GDT on all of the problems except PIND, WINE and HRT. We would expect HHCART(G) to perform better or similar to GDT on most problems because HHCART(G) considers many possible splits in a reflected feature space that is aligned with the two possible splits (the angle bisectors) that GDT considers. Furthermore, HHCART(G)

Table 2. Results of HHCART(G) and other decision tree methods. The reported accuracies and tree sizes (number of leaf nodes) are averages obtained from ten 5-fold cross-validations. Standard deviations are also reported. The highest accuracy for each problem is in bold.

Dataset	Method	Accuracy	Tree Size	Dataset	Method	Accuracy	Tree Size
BS	HHCART(G)	91.9 ± 1.0	15.4 ± 1.0	PIND	HHCART(G)	75.1 ± 1.1	8.2 ± 1.0
	HHCART(A)	$\mathbf{92.8} \pm 1.3$	7.4 ± 1.3		HHCART(A)	73.2 ± 1.4	11.9 ± 6.5
	HHCART(D)	88.3 ± 1.7	12.1 ± 3.3		HHCART(D)	73.7 ± 1.5	11.5 ± 8.4
	OC1	91.9 ± 0.9	8.7 ± 3.4		OC1	73.4 ± 1.0	9.2 ± 5.4
	OC1-AP	78.2 ± 1.3	37.5 ± 16.8		OC1-AP	73.6 ± 1.4	15.9 ± 8.7
	OC1-LC	84.3 ± 1.5	12.6 ± 6.5		OC1-LC	72.8 ± 1.8	11.4 ± 9.6
	GDT	91.8 ± 0.8	20.2 ± 4.1		GDT	$\mathbf{76.7} \pm 0.5$	2 ± 0.0
BH	HHCART(G)	83.3 ± 1.3	9.5 ± 1.1	WINE	HHCART(G)	95.1 ± 0.9	3.8 ± 0.2
	HHCART(A)	$\mathbf{83.4} \pm 1.2$	7.0 ± 2.9		HHCART(A)	91.4 ± 1.8	3.4 ± 0.3
	HHCART(D)	82.0 ± 1.1	8.0 ± 2.8		HHCART(D)	88.3 ± 1.8	4.7 ± 0.7
	OC1	82.2 ± 1.2	9.3 ± 3.4		OC1	89.2 ± 2.1	3.5 ± 0.3
	OC1-AP	82.0 ± 0.7	13.0 ± 5.3		OC1-AP	89.2 ± 4.6	4.6 ± 0.6
	OC1-LC	81.5 ± 1.3	10.6 ± 6.0		OC1-LC	89.4 ± 2.7	3.8 ± 0.6
	GDT	82.2 ± 0.9	33.2 ± 1.6		GDT	$\mathbf{95.8} \pm 1.0$	3.6 ± 0.3
BC	HHCART(G)	$\mathbf{97.2} \pm 0.3$	2.0 ± 0.0	SUR	HHCART(G)	73.0 ± 1.0	2.5 ± 0.2
	HHCART(A)	97.0 ± 0.3	2.3 ± 0.4		HHCART(A)	72.5 ± 1.7	6.5 ± 2.6
	HHCART(D)	97.0 ± 0.3	2.6 ± 1.1		HHCART(D)	72.2 ± 2.2	10.6 ± 5.5
	OC1	95.4 ± 0.5	3.3 ± 1.4		OC1	71.0 ± 2.1	6.4 ± 3.5
	OC1-AP	94.0 ± 0.8	8.3 ± 3.3		OC1-AP	71.9 ± 1.5	10.7 ± 6.5
	OC1-LC	95.5 ± 0.6	3.4 ± 1.6		OC1-LC	70.2 ± 2.4	8.1 ± 4.4
	GDT	95.2 ± 0.5	6.7 ± 1.3		GDT	$\mathbf{73.3} \pm 0.7$	3.4 ± 0.5
BUPA	HHCART(G)	67.2 ± 1.8	7.7 ± 1.2	HRT	HHCART(G)	$\mathbf{84.3} \pm 0.9$	2.0 ± 0.0
	HHCART(A)	64.9 ± 3.0	7.8 ± 1.5		HHCART(A)	75.0 ± 2.3	5.5 ± 1.9
	HHCART(D)	64.8 ± 2.1	10.2 ± 3.0		HHCART(D)	75.2 ± 3.6	8.1 ± 3.1
	OC1	66.9 ± 2.2	8.9 ± 6.1		OC1	77.1 ± 2.5	3.6 ± 1.0
	OC1-AP	64.7 ± 2.5	13.2 ± 10.5		OC1-AP	76.3 ± 2.3	6.7 ± 2.4
	OC1-LC	64.4 ± 2.4	8.9 ± 3.6		OC1-LC	76.3 ± 2.5	4.0 ± 1.1
	GDT	$\mathbf{68.1} \pm 1.4$	15.7 ± 1.4		GDT	82.7 ± 0.8	2.0 ± 0.0
GLS	HHCART(G)	$\mathbf{67.4} \pm 2.3$	11.3 ± 1.1	LET	HHCART(G)	85.3 ± 0.2	1401 ± 11
	HHCART(A)	61.9 ± 3.1	8.8 ± 3.1		HHCART(A)	82.1 ± 0.3	759 ± 88
	HHCART(D)	61.7 ± 3.4	10.7 ± 2.7		HHCART(D)	83.1 ± 0.3	1136 ± 122
	OC1	61.1 ± 3.5	10.8 ± 4.3		OC1	83.6 ± 0.4	1197 ± 89
	OC1-AP	64.6 ± 3.9	14.6 ± 8.7		OC1-AP	$\mathbf{86.3} \pm 0.3$	1612 ± 60
	OC1-LC	$\mathbf{67.4} \pm 2.0$	12.0 ± 3.6		OC1-LC	84.5 ± 0.2	1333 ± 146
	GDT	57.1 ± 2.5	29.3 ± 2.5		GDT	84.8 ± 0.2	2480 ± 19

finds its best split using all available classes and GDT only uses its imposed binary classes. Thus, in terms of accuracy and tree size, HHCART(G) was an effective alternative to GDT.

The average accuracy of HHCART(G) was significantly better than HHCART(A) on BUPA, GLS, PIND, WINE, HRT and LET. For the other four problems, HHCART(A) was not significantly better. For HHCART(D), the average accuracy of HHCART(G) was better on all of the problems, but it was not significantly better on BC and SUR. In terms of average tree size, HHCART(G) produced trees that were similar to the other HHCART trees with smaller standard deviations. These results are important because most of the computational effort for the HHCART approaches goes into forming and searching the reflected feature spaces. HHCART(G) minimises this effort because it only searches p dimensions in one reflected feature space to split each node. HHCART(A) and HHCART(D) search p dimensions in Cp

and C reflected feature spaces, respectively, to split each node (Wickramarachchi et al. 2016). Hence, HHCART(A) and HHCART(D) require at least twice the computational effort of HHCART(G) to split each node. In some cases, the computational advantage of HHCART(G) can be substantially more. On the LET dataset, for example, HHCART(A) and HHCART(D) search 6,656 and 416 splitting dimensions, respectively, and HHCART(G) only searches 16. These computational gains would also make HHCART(G) a more efficient base classifier for bagging (Breiman 1996) or random forests (Breiman 2001). However, a fuller discussion on ensemble methods is beyond the scope of this paper.

We conclude our analysis by making comparisons among the algorithms on the ten datasets considered. To begin, we use the Friedman test (Friedman 1940) with the Iman-Davenport extension (Iman & Davenport 1980) to decide whether all the algorithms have the same performance. The Friedman test is a non-parametric omnibus test that compares the average ranks of the algorithms, $R_k = (1/N)\sum_{i=1}^{N} R_{ik}$, where R_{ik} is the rank of the kth algorithm on the ith dataset (the best performing algorithm has rank one) with $i = 1, \ldots, N$ and $k = 1, \ldots, K$. The Iman-Davenport extension statistic is

$$F_F = \frac{(N-1)\chi_F^2}{N(K-1) - \chi_F^2}$$

with

$$\chi_F^2 = \frac{12N}{K(K+1)}\left(\sum_{k=1}^{K} R_k^2 - \frac{K(K+1)^2}{4}\right),$$

which follows an F distribution with $(K-1)$ and $(K-1)(N-1)$ degrees of freedom. Based on this test, we reject the null hypothesis that all the algorithms have the same performance (p-value = 0.0005) and proceed make pairwise comparisons using post-hoc tests.

The post-hoc statistic for a pairwise comparison between algorithms i and k is

$$Z_F = \frac{R_i - R_k}{\sqrt{\frac{K(K+1)}{6N}}}$$

which follows a standard normal distribution. To compare HHCART(G) with each algorithm, $K-1$ post-hoc tests were performed. To control the family-wise error, the Holm step down procedure (Holm 1979) was used. Let $p_1 \leq p_2 \leq \ldots \leq p_{K-1}$ denote the sequence of ordered p-values with corresponding hypotheses $H_1, \ldots, H_{K-1}$. We reject hypotheses $H_1, \ldots, H_{j-1}$ at level α where $j = \min\{j : p_j > \alpha/(K-j)\}$. At level $\alpha = 0.05$, HHCART(G) was significantly better than HHCART(D), OC1, OC1-AP and OC1-LC, but it was not significantly better than HHCART(A) (p-value = 0.038) and GDT (p-value = 0.161).

6. Conclusion

In this paper, we have presented an algorithm for learning oblique decision trees. HHCART(G) uses a modified GDT angle bisector to define a reflected feature space for an HHCART decision tree. At each node, we reflect the training examples with respect to the angle bisector to align linear structure in the training examples with the coordinate axes. Searching axis parallel splits in this reflected feature space provides an efficient and effective way of finding oblique splits in the original feature space. HHCART(G) is much simpler and more efficient to build than the other HHCART methods because it only uses one reflected feature space to split each node. HHCART(A) and HHCART(D) consider Cp and C reflected feature spaces, respectively, to split each node making them more computationally intensive to build. Experimental results showed that HHCART(G) was an effective classifier, producing compact trees with similar or better results than several other decision trees. HHCART(G) performed better than GDT on most of the datasets. Its average accuracy was significantly better on six datasets and GDT was only significantly better on one dataset. HHCART(G) trees were also more compact than GDT trees, with average tree sizes that were significantly smaller on seven datasets. The average accuracy of HHCART(G) was significantly better than HHCART(A) on six datasets and HHCART(A) was not significantly better on the remaining four datasets. When compared with HHCART(D), the average accuracy of HHCART(G) was better on all of the datasets considered. Hence, HHCART(G) is a computationally efficient alternative to both HHCART methods. The methods were also compared on all ten datasets using a Friedman test. HHCART(G) was significantly better than HHCART(D), OC1, OC1-AP and OC1-LC, but it was not significantly better than HHCART(A) and GDT.

References

AMASYAH, M. & ERSOY, O. (2008). Cline: A new decision-tree family. *Neural Networks, IEEE Transactions on* **19**, 356–363.

BACHE, K. & LICHMAN, M. (2013). UCI machine learning repository. URL `http://archive.ics.uci.edu/ml`.

BERTSIMAS, D. & DUNN, J. (2017). Optimal classification trees. *Machine Learning* **106**, 1039–1082.

BREIMAN, L. (1996). Bagging predictors. *Machine Learning* **24**, 123–140.

BREIMAN, L. (2001). Random forests. *Machine Learning* **45**, 5–32.

BREIMAN, L., FRIEDMAN, J., STONE, C.J. & OLSHEN, R.A. (1984). *Classification and Regression Trees.* CRC Press, New York.

CANTÚ-PAZ, E. & KAMATH, C. (2003). Inducing oblique decision trees with evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* **7**, 54–68.

FRIEDMAN, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics* **11**, 86–92.

GAMA, J. & BRAZDIL, P. (1999). Linear tree. *Intelligent Data Analysis* **3**, 1–22.

GOLUB, G.H. & VAN LOAN, C.F.V. (1996). *Matrix Computations*, vol. 3rd ed. Baltimore, Maryland, USA: John Hopkins University Press.

HASTIE, T., TIBSHIRANI, R. & FRIEDMAN, J. (2013). *The Elements of Statistical Learning*, vol. 2nd ed. New York, New York, USA: Springer.

HEATH, D., KASIF, S. & SALZBERG, S. (1993). Induction of oblique decision trees. *Journal of Artificial Intelligence Research* **2**, 1–32.

HOLM, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics* **6**, 65–70.

IMAN, R. & DAVENPORT, J. (1980). Approximation of the critical region of the Friedman statistic. *Communications in Statistics - Theory and Methods* **9**, 571–595.

KOLAKOWSKA, A. & MALINA, W. (2005). Fisher sequential classifiers. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* **35**, 988–998.

LI, X.B., SWEIGART, J.R., TENG, J.T., DONOHUE, J.M., THOMBS, L.A. & WANG, S.M. (2003). Multivariate decision trees using linear discriminants and tabu search. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* **33**, 194–205.

LI, Y., DONG, M. & KOTHARI, R. (2005). Classifiability-based omnivariate decision trees. *IEEE Transactions on Neural Networks* **16**, 1547–1560.

LÓPEZ-CHAU, A., CERVANTES, J., LÓPEZ-GARCÍA, L. & LAMONT, F.G. (2013). Fisher's decision tree. *Expert Systems with Applications* **40**, 6283–6291.

MANWANI, N. & SASTRY, P.S. (2012). Geometric decision tree. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* **42**, 181–192.

MURTHY, S.K., KASIF, S. & SALZBERG, S. (1993). The OC1 decision tree software system. URL `http://www.cbcb.umd.edu/salzberg/announce-oc1.html`.

MURTHY, S.K., KASIF, S. & SALZBERG, S. (1994). A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research* **2**, 1–32.

RIVERA-LOPEZ, R., CANUL-REICH, J., GÁMEZ, J.A. & PUERTA, J.M. (2017). OC1-DE: A Differential Evolution Based Approach for Inducing Oblique Decision Trees. In *Artificial Intelligence and Soft Computing*, eds. L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L.A. Zadeh & J.M. Zurada. Cham: Springer International Publishing, pp. 427–438.

ROBERTSON, B.L., PRICE, C.J. & REALE, M. (2013). CARTopt: a random search method for nonsmooth unconstrained optimization. *Computational Optimization and Applications* **56**, 291–315.

SHEIKHOLHARAM MASHHADI, P. (2018). Boosted Test-FDA: a transductive boosting method. *Pattern Analysis and Applications* doi:10.1007/s10044-018-0710-7. URL `https://doi.org/10.1007/s10044-018-0710-7`.

TRUONG, A. (2009). Fast growing and interpretable oblique trees via logistic regression models. Ph.D. thesis, University of Oxford.

WICKRAMARACHCHI, D.C. (2015). Oblique decision trees in transformed spaces. Ph.D. thesis, School of Mathematics and Statistics, University of Canterbury.

WICKRAMARACHCHI, D.C., ROBERTSON, B.L., REALE, M., PRICE, C.J. & BROWN, J.A. (2016). HHCART: An oblique decision tree. *Computational Statistics and Data Analysis* **96**, 12–23.